Iterative Soft-Decision Reed-Solomon Decoding on Partial Response Channels

Michael K. Cheng and Paul H. Siegel
Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407

***Abstract*— Since the discovery of Turbo Codes, iterative decoding has gained enormous momentum. The idea of repeatedly passing information between components of a receiver or decoder to increase the overall system performance has attracted much research effort. In this work, we present an iterative soft-decision decoding architecture for Reed-Solomon (RS) codes on a partial-response (PR) channel. The architecture incorporates a symbol-based a-posteriori probability (APP) detector for the channel, and an enhanced soft-decision RS decoder based upon the recently introduced Koetter-Vardy (KV) algorithm. From the list of candidate RS codewords generated by the KV decoder, we calculate output symbol reliabilities that can be fed back to the APP detector as extrinsic information to be used in a subsequent decoding iteration. Through simulations, we show the efficacy of this approach, especially when the initial KV list size is large. We will also propose ways to modify the decoding scheme in order to beneficially increase the size of the candidate codeword list and thereby improve the overall system performance.**

I. Introduction

The discovery of Turbo codes by Berrou, Glavieux, and Thitimajshima [1] in 1993 has generated much interest in iterative decoding. Turbo decoding consists of two Soft-Input Soft-Output (SISO) decoders passing information between one another to iteratively refine the overall decoding decision. We modify the one-pass soft-decision RS decoding scheme for PR channels, shown in Figure 1, to include iterations. Through repeated decoding, we aim to improve the accuracy of the *a-priori* probability input to the symbol-based APP detector.

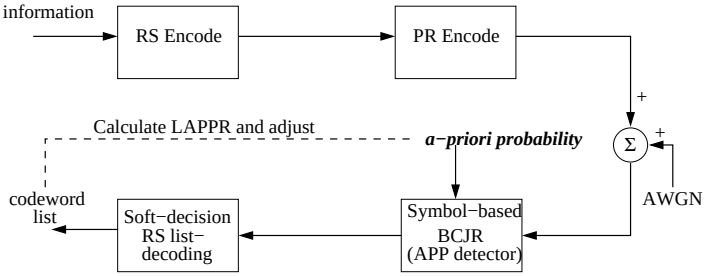

Fig. 1. The conventional or one-pass Reed-Solomon (RS) decoding architecture. The dashed line marks the proposed modification to run iterative decoding.

We are motivated by Pyndiah's approach [2] that generates the log likelihood ratio (LLR) from a list of codewords for each symbol position. Pyndiah's LLR description is actually the log ratio of two *a-posteriori* probabilities (APPs), therefore, a more appropriate term for Pyndiah's LLR is the log *a-posteriori* probability ratio (LAPPR) [3]. We use the extrinsic portion of the LAPPR to adjust the *a-priori* probabilities in the symbol-based BCJR algorithm [4] for re-decoding. The recalculated symbol reliabilities will allow the soft-decision RS decoder to refine the candidate codeword list and improve the decoding performance.

This work was supported by the Information Storage Industry Consortium (INSIC) and by the Center for Magnetic Recording Research (CMRR), University of California San Diego.

In Section II we discuss Pyndiah's iterative turbo decoding of product codes. In Section III we review the conventional one-pass soft-decision RS decoding architecture for PR channels. We then describe how to modify the conventional architecture and extend Pyndiah's LAPPR calculation to allow iterative decoding. In Section IV we provide the details of the LAPPR calculation and discuss the need to weight the feedback extrinsic information. In Section V we present simulation results that confirm the efficacy of the iterative soft-decision RS decoding scheme. In Section VI we discuss methods to further improve the iterative RS decoding performance. In Section VII we conclude with some remarks.

II. Iterative Turbo Decoding of Product Codes

The product code, shown in Figure 2, consists of a (n_1, k_1)

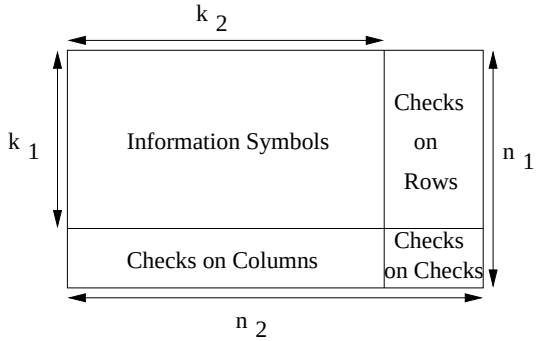

Fig. 2. Construction of a product code $\mathcal{C}_1 \times \mathcal{C}_2$. The code parameters are n, the length, and, k, the dimension.

column code and a (n_2, k_2) row code. Pyndiah in [2] and [5] considers the transmission of a binary product code over the Gaussian channel. At the receiver end, each component code is decoded using the Chase algorithm [6] and a list of candidate codewords is generated for each decoding attempt. Let $\underline{c} = (c_1, \cdots, c_j, \cdots, c_n)$ be the transmitted row ($n = n_2$) or column ($n = n_1$) codeword, $\underline{y} = (y_1, \cdots, y_j, \cdots, y_n)$ be the codeword plus additive white Gaussian noise (AWGN) with variance σ^2, and $\underline{d} = (d_1, \cdots, d_j, \cdots d_n)$ be the decision vector. The log *a-posteriori* probability ratio (LAPPR) of the decision d_j is defined by

$$\Lambda(d_j) = \ln\left(\frac{Pr\left\{c_j = +1 \mid \underline{y}\right\}}{Pr\left\{c_j = -1 \mid \underline{y}\right\}}\right) \tag{1}$$

The numerator can be expanded as

$$Pr\left\{c_j = +1 \mid \underline{y}\right\} = \sum_{\underline{c} \in S_j^{+1}} Pr\left\{\underline{c} \mid \underline{y}\right\} \tag{2}$$

where S_j^{+1} is the set of codewords in the Chase decoded list such that $c_j = +1$. The denominator can be expanded over the codewords in the similarly defined set S_j^{-1}. Using Bayes' rule and assuming equally likely codewords we can rewrite the LAPPR as

$$\Lambda(d_j) = \ln\left(\frac{\sum_{\underline{c}\in S_j^{+1}} p\left\{\underline{y} \mid \underline{c}\right\}}{\sum_{\underline{c}\in S_j^{-1}} p\left\{\underline{y} \mid \underline{c}\right\}}\right) \tag{3}$$

where

$$p\left\{\underline{y} \mid \underline{c}\right\} = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{\left|\underline{y}-\underline{c}\right|^2}{2\sigma^2}\right) \tag{4}$$

Pyndiah showed [2] that for high SNR, the LAPPR of (3) can be approximated as

$$\Lambda(d_j) \approx \frac{2}{\sigma^2}(y_j + w_j) \tag{5}$$

and w_j is the extrinsic information for the symbol position j. To obtain w_j, we can calculate the LAPPR from the Chase output using (3) and subtract the channel value from the weighted LAPPR, i.e.

$$w_j \approx \frac{\sigma^2}{2}\Lambda(d_j) - y_j \tag{6}$$

We can calculate the extrinsic information for each entry of the product code matrix and use this information to modify the channel value for re-decoding. Let $[\mathbf{W}(m)]$ be the calculated extrinsic information of the m^{th} iteration and $[\mathbf{Y}]$ be the received channel matrix (see Figure 2). Then the channel value for the m^{th} iteration is given by

$$[\mathbf{Y}(m)] = [\mathbf{Y}] + \alpha(m)[\mathbf{W}(m)] \tag{7}$$

and $\alpha(m)$ is a weighting factor to reduce the effects of the feedback information $[\mathbf{W}(m)]$, especially in the early iterations when the bit error rate is high. Pyndiah's turbo product decoder is illustrated in Figure 3.

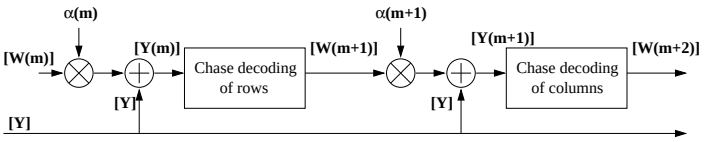

Fig. 3. Pyndiah's iterative turbo decoding of product codes.

III. Modifying soft-decision RS decoding on PR channels to include iterations

We consider an (n, k, d) RS code over $GF(q)$, where d is the minimum distance, $q = 2^l$, and there are l bits per field symbol. The one-pass soft-decision RS decoding on PR channels is depicted in Figure 1. The system consists of a symbol-based BCJR which calculates the symbol-wise reliabilities for each l-bit symbol [4] and a Koetter-Vardy decoder [7] which takes as input symbol-wise reliabilities and outputs a list of candidate codewords. We would like to modify the configuration in Figure 1 and incorporate symbol-based LAPPR calculation in order to apply iterative decoding. The idea, illustrated in Figure 4, is to include an LAPPR computation block that takes as input the KV

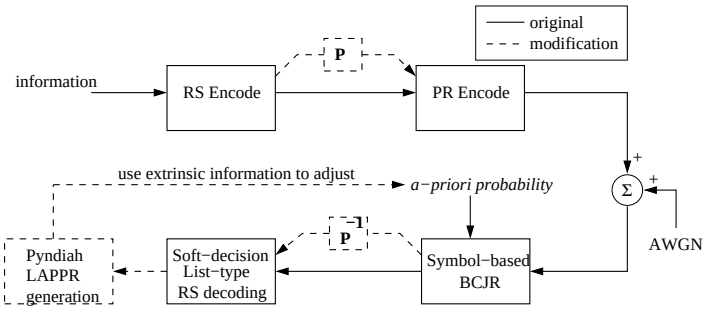

Fig. 4. Modifying the one-pass soft-decision RS decoding architecture for PR channels to implement iterative decoding.

codeword list and generates the LAPPR for each symbol position. We can then extract the extrinsic information for each symbol and use the extrinsic information to adjust the *a-priori* probabilities input to the symbol-based BCJR algorithm. Since the first pass of the symbol-based BCJR uses equal *a-priori* probability for each symbol, the re-adjusted prior probabilities will "pin" each symbol path in the PR trellis with a corresponding likelihood value (see Figure 5) much like the state pinning technique introduced by Collins [8]. An interleaver is added to enhance the symbol pinning effect, i.e., if a correct codeword is found by the KV algorithm and its symbols are pinned by the feedback procedure, the interleaver will distribute the correct symbols in the PR trellis to break up burst error patterns. The KV decoder and the symbol-based BCJR work together symbiotically to improve the overall decoding performance.

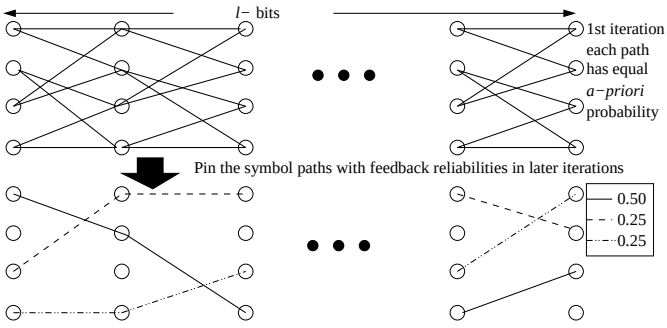

Fig. 5. Pinning the symbol paths in the PR trellis with likelihood values and removing those paths that cannot occur.

IV. Iterative soft-decision RS decoding on PR channels

The iterative soft-decision RS decoding architecture for PR channels is shown in Figure 6. We include a block interleaver after the RS encoder. The codewords are written row-wise into and read column-wise out of the interleaver. The permuted symbols are then PR encoded and transmitted over the AWGN channel. To decode the received channel values, we apply the symbol-based BCJR algorithm to generate the symbol reliabilities and run the KV algorithm to produce a candidate codeword list. We calculate the LAPPR for each symbol position from the codeword list, extract the extrinsic information, weight and feedback the permuted information to adjust the *a-priori* probabilities input

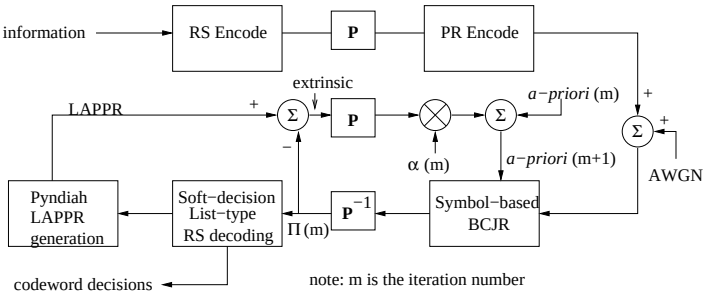

Fig. 6. Iterative Soft-Decision Reed-Solomon Decoding on Partial Response Channels.

to the symbol-based BCJR algorithm. We continue the iterations until the desired number has been reached.

A. Generating the log a-posteriori probability ratio (LAPPR) from a list of codewords

For an (n, k) RS code, the symbol-based BCJR will output a symbol reliability matrix of the following form for each channel vector :

$$\Pi = \begin{bmatrix} \pi_{1,1} & \pi_{1,2} & \cdots & \pi_{1,n-1} & \pi_{1,n} \\ \pi_{2,1} & \pi_{2,2} & \cdots & & \pi_{2,n} \\ \vdots & & \ddots & & \vdots \\ \pi_{q-1,1} & & & & \pi_{q-1,n} \\ \pi_{q,1} & & \cdots & & \pi_{q,n} \end{bmatrix} \tag{8}$$

where each row index represents a field element $\alpha_i \in GF(q)$ and each column index represents a symbol position in the codeword. Each entry of Π is defined as

$$\pi_{i,j} = Pr\left(c_j = \alpha_i \mid \underline{y}\right), \quad i = 1, 2, \cdots, q, \text{ and, } j = 1, 2, \cdots, n \tag{9}$$

For each KV codeword list $\{\underline{c}\}$ and its corresponding reliability Π, we want to generate another size $q \times n$ matrix, Π^{LAPPR}, whose entries are

$$\pi_{i,j}^{\text{LAPPR}} = \ln\left(\frac{\sum_{\underline{c} \in S_j^i} Pr\left\{\underline{c} \mid \Pi\right\}}{\sum_{\underline{c} \in \bar{S}_j^i} Pr\left\{c \mid \Pi\right\}}\right) \tag{10}$$

where $S_j^i = \{\underline{c} \in \text{K-V output list} : c_j = \alpha_i\}$ is the set of codewords in the KV list such that the j^{th} symbol position is α_i and $\bar{S}_j^i = \{\underline{c} \in \text{K-V output list} : c_j \neq \alpha_i\}$ is the set of codewords in the KV list such that the j^{th} symbol position is not α_i. We can express the conditional probability in (10) as

$$Pr\left\{\underline{c} \mid \Pi\right\} = \prod_{v=1}^{n} \pi_{u,v}; \quad \{u \in [1, 2, \cdots, q] : c_v = \alpha_u\} \tag{11}$$

Substituting (11) into (10), we have

$$\begin{aligned} \pi_{i,j}^{\text{LAPPR}} &= \ln\left(\frac{\sum_{\underline{c} \in S_j^i} \pi_{i,j} \cdot \prod_{v=1, v \neq j, c_v = \alpha_u}^{n} \pi_{u,v}}{\sum_{\underline{c} \in \bar{S}_j^i} \prod_{v=1, c_v = \alpha_u}^{n} \pi_{u,v}}\right) \\ &= \ln\left(\pi_{i,j}\right) + \ln\left(\frac{\sum_{\underline{c} \in S_j^i} \prod_{v=1, v \neq j, c_v = \alpha_u}^{n} \pi_{u,v}}{\sum_{\underline{c} \in \bar{S}_j^i} \prod_{v=1, c_v = \alpha_u}^{n} \pi_{u,v}}\right) \\ &= \ln\left(\pi_{i,j}\right) + w_{i,j} \end{aligned} \tag{12}$$

where $w_{i,j}$ is the extrinsic information to be used to adjust the *a-priori* probability input to the symbol-based BCJR algorithm.

B. The size of the KV decoded list versus the code rate.

The size of the KV decoded list is a function of the code rate. The accuracy of the LAPPR depends on the list size. The larger the codeword list, the better the LAPPR quality. The performance of Guruswami and Sudan's list-decoding (upon which the KV algorithm is based) versus the rate of the RS code is plotted in [9, Figure 1]. We see that the size of the list is dependent on the RS code rate because the higher the rate, the smaller the error radius, and the decoded list will contain fewer codewords. Moreover, for high rate codes, the list-decoding capability converges to that of classical (half the minimum distance) decoding and list-decoding will most likely output one or zero codewords. If the list contains only an incorrect codeword, the feedback information may decrease the performance of the next iteration. For these reasons, the feedback information should be weighted before being used to adjust the *a-priori* probabilities.

C. Weighting the feedback extrinsic information

The weighting factor $0 \leq \alpha(m) \leq 1$ for the m^{th} iteration is optimized through simulations and may be different for each iteration. The idea is to have a low $\alpha(m)$ in the initial iterations when the error rate is high and increase its magnitude as the quality of the feedback information improves from trial-to-trial. For a specific simulation run, the Word Error Rate (WER) may fluctuate. This is especially true for high rate codes. On average, however, iterations will improve the overall performance.

D. Steps to iterative soft-decision RS decoding

Initialization Step: Let $m = 0$ be the iteration number and set the *a-priori* probability of each symbol to be equal, i.e. let the *a-priori* probability matrix be $\Pi^{a-priori}(m)$ and set $\pi_{i,j}^{a-priori}(0) = 1/q$ for $i = 1, 2, \cdots, q$ and $j = 1, 2, \cdots, n$.

Iteration Steps:

1) Apply the symbol-based BCJR algorithm to the channel values along with $\Pi^{a-priori}(m)$ to calculate the reliability matrix $\Pi(m)$.
2) Apply the Koetter-Vardy soft-decision RS decoding using input $\Pi(m)$.
3) Apply Pyndiah's method to the KV list to generate the $q \times n$ matrix $\Pi^{\text{LAPPR}}(m)$.
4) Calculate the extrinsic information by: $W^{\text{extrinsic}}(m) = \Pi^{\text{LAPPR}}(m) - \ln(\Pi(m))$.
5) Adjust the *a-priori* probability for the next pass of the symbol-based BCJR algorithm by: $\Pi^{a-priori}(m+1) = \Pi^{a-priori}(m) + \alpha(m) \cdot W^{\text{extrinsic}}(m)$.
6) Increment m by one and repeat.

V. Simulation Results

The Guruswami-Sudan (GS) list-decoding involves finding a bi-variate polynomial that interpolates a subset of the points in a length n received word [9], [10]. The polynomial will pass through each fitted point with a multiplicity m. Obtaining the coefficients of the interpolating polynomial requires solving a system of linear equations. The number of equations in the system, also referred to as the cost, is given by

$$C = n\left(m\left(m+1\right)/2\right). \tag{13}$$

Iterations\SNR	0 dB	1 dB	2 dB	3 dB	4 dB
0	0.5240	0.2790	0.0830	0.0132	0.0007
1	0.4440	0.2070	0.0410	0.0022	0
2	0.3900	0.1650	0.0310	0.0013	0
3	0.3680	0.1440	0.0260	0.0012	0
4	0.3470	0.1340	0.0220	0.0011	0
5	0.3400	0.1270	0.0220	0.0011	0

TABLE I

WER, SNR, AND ITERATION COUNT OF A $(15, 7, 9)$ RS CODE OVER THE EPR4 CHANNEL AND AVERAGED OVER $10{,}000$ CODEWORDS.

Iterations\SNR	3 dB	4 dB	5 dB
0	0.1140	0.0171	0.0013
1	0.0578	0.0039	0.0001
2	0.0537	0.0035	0.0001
3	0.0535	0.0035	0.0001
4	0.0534	0.0035	0.0001
5	0.0534	0.0035	0.0001

TABLE II

WER, SNR, AND ITERATION COUNT OF A $(15, 9, 7)$ RS CODE OVER THE EPR4 CHANNEL AND AVERAGED OVER $10{,}000$ CODEWORDS.

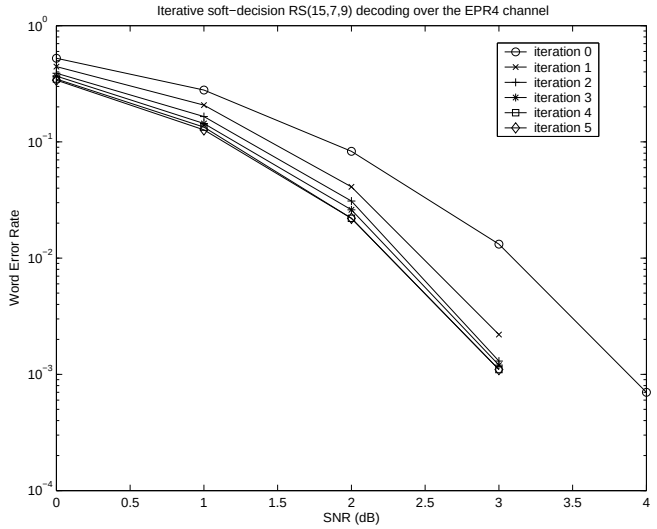

Fig. 7. WER vs SNR plot of a $(15, 7, 9)$ RS code over the EPR4 channel and averaged over 10,000 codewords

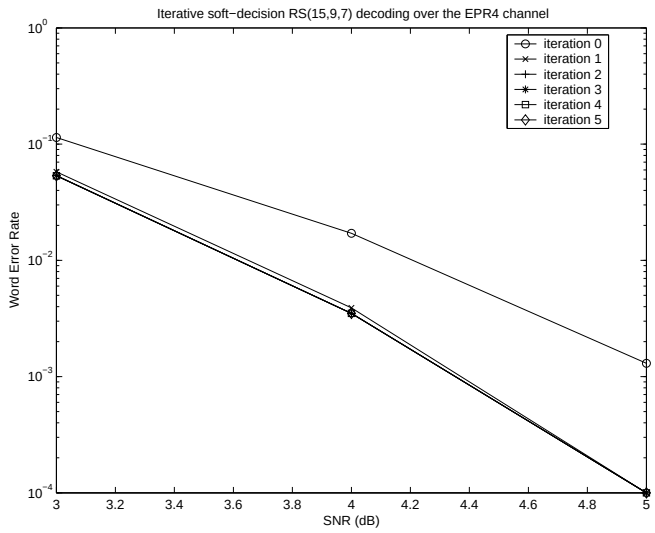

Fig. 8. WER vs SNR plot of a $(15, 9, 7)$ RS code over the EPR4 channel and averaged over 10,000 codewords.

Therefore, the complexity of the GS list-decoding can be high, especially for long codewords and large multiplicities. Because the KV algorithm is based on the GS interpolation technique, KV decoding can also have a high computational cost . To lower the complexity of the GS and the KV algorithms, Gross and others have proposed a re-encoding technique [11] that decreases the number of equations to be solved from C to

$$C' = (n-k)\left(m\left(m+1\right)/2\right) = \left(1-k/n\right)C. \qquad (14)$$

The saving in the cost increases as a function of the code rate k/n.

To reduce simulation time, we apply soft-decision iterative decoding to Reed-Solomon (RS) codes defined over $GF\left(16\right)$. We would expect the decoding behavior to extend to RS codes defined over higher fields. We use the EPR4 channel with transfer function $h\left(D\right) = \left(1-D\right)\left(1+D\right)^2$ as the inner code. The first simulation uses a $(15, 7, 9)$ RS code and consists of decoding 10 trials of 1000 codewords each. The result is, therefore, averaged over 10,000 codewords. The extrinsic information weighting factors for the iterations are $\alpha\left(m\right) = [0.01, 0.01, 0.01, 0.02, 0.1]$. We increase the weights slowly as the LAPPR becomes more accurate with iterations. The Word Error Rate (WER) versus the channel SNR is given in Table I and plotted in Figure 7. We see that at a WER of 1×10^{-2} there is a 0.5 dB gain after 1 iteration and about 0.75 dB gain after 4.

The second simulation uses a $(15, 9, 7)$ RS code and consists of, again, averaging 10 trials of 1000 codewords. We use a different weighting vector $\alpha\left(m\right) = [0.05, 0.06, 0.07, 0.1, 0.2]$. The WER versus SNR is given in Table II and plotted in Figure 8. After 1 iteration there is a SNR gain of 0.6 dB at WER of 1×10^{-3}. Applying simply one iteration of soft-decision RS decoding can provide a noticeable performance improvement.

VI. METHODS TO IMPROVE THE PERFORMANCE OF ITERATIVE SOFT-DECISION RS DECODING

A. Exhaustive list-generation

Because the list-size can be small for high-rate codes, we consider an exhaustive approach to generating a larger codeword list that would lead to a more accurate LAPPR calculation. The method involves pinning each entry $\pi_{i,j}$ of the reliability matrix Π to 1 and applying the KV algorithm to obtain a list of candidate codewords that contain the i^{th} symbol at the j^{th} position. For example, if we want to generate a list of candidate codewords that have the symbol α_2 in the second position, we can set the $(2, 2)$ entry of the reliability matrix Π to 1 and zero out the remaining elements in column 2. We would, therefore, input the following reliability matrix to the KV algorithm:

$$\Pi = \begin{bmatrix} \pi_{1,1} & 0 & \cdots & \pi_{1,n-1} & \pi_{1,n} \\ \pi_{2,1} & 1 & \cdots & & \pi_{2,n} \\ \vdots & \vdots & \ddots & & \vdots \\ \pi_{q-1,1} & 0 & \pi_{i,j} & & \pi_{q-1,n} \\ \pi_{q,1} & 0 & \cdots & & \pi_{q,n} \end{bmatrix}. \qquad (15)$$

$u \to v$	No exhaustive list-generation	Chase-type list-generation
$0 \to 1$	-0.719 dB	-2.578 dB
$0 \to 2$	-1.283 dB	-3.316 dB
$0 \to 3$	-1.535 dB	-3.698 dB
$0 \to 4$	-1.790 dB	-3.935 dB
$0 \to 5$	-1.879 dB	-4.232 dB

TABLE III

THE WERR COMPARISON OF "CHASE" PINNING THE 7 LEAST RELIABLE COLUMNS OF A $(15, 7, 9)$ RS CODE ON THE EPR4 CHANNEL AT 0 DB SNR.

We repeat the process for all $q \times n$ entries and take the union of the lists generated by each KV decoding to increase the overall candidate list size. The complexity is, however, high.

To reduce the computation requirement, we can use a "Chase" variation [6] in which we only pin the least reliable s columns, $s \leq n$. Reliability can be determined by taking the difference between the maximum entry and the minimum entry of a column. To decide which columns to pin, we first calculate

$$\pi_j^{\Delta} = \pi_{i,j}^{max} - \pi_{i,j}^{min}, \tag{16}$$

where $\pi_{i,j}^{max}$ is the largest entry and $\pi_{i,j}^{min}$ is the smallest entry of column j. Then we sort the π_j^{Δ}'s and only pin the s columns that have the smallest s values. A column with a large π_j^{Δ} can be interpreted as reliable because the KV Algorithm A [7] will concentrate its multiplicity allocation in only a few entries in that column, whereas, a column with a small π_j^{Δ} can be regarded as unreliable because the proximity of the values of the column entries will cause Algorithm A to spread the multiplicities to many entries in that column. Since the total number of multiplicities to be used is fixed, KV decoding is more certain to find a list of candidate codewords when the multiplicities are concentrated in a few locations as opposed to being spread out across the reliability matrix Π. Using the "Chase" variation, we only apply $q \times s$ instances of KV decoding instead of $q \times n$. Let us define the Word Error Rate Ratio (WERR) as

$$WERR(u, v) = 10 \cdot \log_{10}\left(\frac{WER_v}{WER_u}\right), \tag{17}$$

which indicates the improvement of WER from iteration u to iteration v. To illustrate the performance and complexity trade-off of increasing the list size, we plot the WERR of a $(15, 7, 9)$ RS code on the EPR4 channel at a SNR of 0 dB in Table III. In this example, we run the KV decoding algorithm $q \times s = 16 \times 7 = 112$ instead of $q \times n = 16 \times 15 = 240$ times for the exhaustive approach. The complexity and performance trade-off can be managed by adjusting the number of columns, s, whose entries are pinned and to which we apply the KV decoding.

B. Interleaving codes of different rates

We can also improve the performance of iterative decoding by periodically inserting a codeword from a lower rate RS code in the transmission. Soft-decision RS decoding of the lower rate code will lead to the correct transmitted symbols with high probability and these symbols can pin the trellis to aid the decoding of higher rate code symbols. As an example, for every three $(15, 7)$ RS codewords, we can insert a $(15, 6)$ RS codeword to improve the overall system performance at a small rate penalty, shown in Table IV.

$u \to v$	No interleaving, rate=0.467	Interleaving, rate=0.45
$0 \to 1$	-0.719 dB	-0.887 dB
$0 \to 2$	-1.283 dB	-1.665 dB
$0 \to 3$	-1.535 dB	-2.002 dB
$0 \to 4$	-1.790 dB	-2.182 dB
$0 \to 5$	-1.879 dB	-2.223 dB

TABLE IV

THE WERR COMPARISON OF INTERLEAVING ONE $(15, 6, 10)$ RS CODEWORD FOR EVERY THREE $(15, 7, 9)$ RS CODEWORDS (EPR4 AT 0DB SNR.)

VII. CONCLUSION

We have presented an approach for applying iterations to soft-decision RS decoding. Our decoding architecture consists of a symbol-based BCJR block, a soft-decision list-decoder, an interleaver, and a LAPPR (log *a-posteriori* probability ratio) generation block. The symbol-based BCJR calculates symbol-wise reliabilities using the Partial Response (PR) trellis of the channel. The soft-decision list-decoder, in this case the Koetter-Vardy algorithm, takes the symbol reliabilities as input and generates a list of candidate codewords. The interleaver removes correlated error patterns and the LAPPR generation block calculates the extrinsic information that can be used to adjust the *a-priori* symbol probability provided to the symbol-based BCJR in re-decoding. The channel decoder (symbol-based BCJR) and the soft-decision list-decoder (the KV algorithm) iteratively exchange soft information with each other to improve the overall decoding performance. Simulation results indicate that a noticeable performance gain can be obtained after the first iteration. Methods to further improve the iterative soft-decision RS decoding performance such as exhaustive list-generation and interleaving of lower rate RS codewords were also discussed.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Conf. Commun.*, vol. 2, (Geneva, Switzerland), pp. 1064–1070, IEEE, May 1993.

[2] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Commun.*, vol. 46, pp. 1003–1010, Aug. 1998.

[3] T. Souvignier, M. Öberg, P. H. Siegel, R. E. Swanson, and J. K. Wolf, "Turbo decoding for partial response channels," *IEEE Trans. Commun.*, vol. 48, pp. 1297–1308, Aug. 2000.

[4] M. K. Cheng, J. Campello, and P. H. Siegel, "Soft-decision Reed-Solomon decoding on partial response channels," in *Proc. IEEE Global Telecom. Conf.*, (Taipei, Taiwan, ROC), IEEE, Nov. 2002.

[5] O. Aitsab and R. Pyndiah, "Performance of concatenated Reed-Solomon convolutional codes with iterative decoding," in *Proc. IEEE Global Telecom. Conf.*, vol. 2, (Phoenix, AZ, USA), pp. 934–938, IEEE, 1997.

[6] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Inform. Theory*, vol. 18, pp. 170–182, Jan. 1972.

[7] R. Kötter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," in *Proc. IEEE Int. Symp. Information Theory*, (Sorrento, Italy), IEEE, June 2000.

[8] O. M. Collins and M. Hizlan, "Determinate state convolutional codes," *IEEE Trans. Commun.*, vol. 41, pp. 1785–1794, Dec. 1993.

[9] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and Algebraic-Geometry codes," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1757–1767, Sept. 1999.

[10] M. Sudan, "Decoding of Reed-Solomon codes beyond the error correction bound," *J. Complexity*, vol. 13, pp. 180–193, Sept. 1997.

[11] W. J. Gross, F. R. Kschischang, R. Koetter, and P. G. Gulak, "A VLSI architecture for interpolation in soft-decision list-decoding of Reed-Solomon codes," in *Proceedings of the 2002 IEEE Workshop on Signal Processing Systems*, (San Diego, CA), pp. 39–44, IEEE, Oct. 2002.